MELJJ

Software Configuration Management Plan

Version 1.0

February 6, 2020

**Document Control**

**Approval**

The Guidance Team and the customer shall approve this document.

**Document Change Control**

|  |  |
| --- | --- |
| Initial Release: | 1.0 |
| Current Release: | 1.3 |
| Indicator of Last Page in Document: | $ |
| Date of Last Review: | 02/06/2020 |
| Date of Next Review: | 02/09/2020 |
| Target Date for Next Update: | 02/18/2020 |

**Distribution List**

This following list of people shall receive a copy of this document every time a new version of this document becomes available:

Guidance Team Members:

Dr. Gates

Dr. Salamah

Dr. Roach

Elsa Tai Ramirez

Jake Lasley

Customer: Dr. Oscar Perez

Vincent Fonseca

Herandy Denisse Vazquez

Baltazar Santaella

Florencia Larsen

Erick De Nava

Software Team Members:

Jazmin Paz

Leslie Gomez

Eduardo Herrera

Micheal Sansone

Jorge Flores

**Change Summary**

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 | 02/06/2020 | Jazmin Paz | Added Section 1 Introduction to SCM |
| 1.1 | 02/06/2020 | Micheal Sansone | Added Section 2 |
| 1.2 | 02/06/2020 | Eduardo Herrera | Added section 3.3 |
| 1.3 | 02/06/2020 | Leslie Gomez | Added Section 3.2 & 4 |

**Table of Contents**

[**DOCUMENT CONTROL II**](#_heading=h.30j0zll)

[**Approval ii**](#_heading=h.44sinio)

[**Document Change Control ii**](#_heading=h.2jxsxqh)

[**Distribution List ii**](#_heading=h.z337ya)

[**Change Summary ii**](#_heading=h.3j2qqm3)

[**1.**](#_heading=h.3dy6vkm) **INTRODUCTION 1**

[**1.1.**](#_heading=h.1t3h5sf) **References 1**

[**2.**](#_heading=h.4d34og8) **SOFTWARE CONFIGURATION IDENTIFICATION 1**

[**2.1.**](#_heading=h.2s8eyo1) **Software Configuration Item Identification 1**

[**2.2.**](#_heading=h.17dp8vu) **Software Configuration Item Organization 1**

[**3.**](#_heading=h.3rdcrjn) **SOFTWARE CONFIGURATION CONTROL 1**

[**3.1.**](#_heading=h.26in1rg) **Documentation 1**

[**3.2.**](#_heading=h.lnxbz9) **Configuration Control Board 2**

[**3.3.**](#_heading=h.35nkun2) **Procedures 2**

[**4.**](#_heading=h.1ksv4uv) **SOFTWARE CONFIGURATION AUDITING 2**

# Introduction

The system to be developed is named the PMR Insight Collective Knowledge (PICK), and it is a tool used to perform analyses that has the capability of deriving a thorough story from log files having to do with a series of events that take place on a given network. The system is able to ingest raw log files of a multitude of formats, parse the information, and store within the system as log entries. Furthermore, the user can specify a vector to analyze, flagging significant events and introducing them as nodes of the vector. Once the vector is finalized, the system can export a visual representation of the vector, known as a graph, as well as the table representation of the vector. The purpose of the system is to facilitate and accelerate event analysis, which would otherwise take up to months to complete. The system will be dynamic in that it will allow for little user intervention, if desired, because most of the work will be performed by the system.

The software configuration management document for the PICK system serves as a means of 1. identifying configuration items (CIs) for the system, 2. controlling modifications and versions of CIs where the configuration control board that will be in charge of approving/disapproving modification to the projet, and software configuration auditing to determine the degree in which the current system under development resembles the system proposed by the stakeholder.

The software configuration identification section describes the list of configuration items that comprise our system. These such CIs include the third-party Extract-Transform-Load (ETL) tool, Splunk, that our system will include, source code, user documentation, etc. Additionally, it describes the organization of the CIs, such as the syntax and naming conventions source code classes will adopt, and the location in which they will be stored. The software configuration control section describes the way in which modifications and versions will be documented and kept track of. This section will also include who will be in charge of modifying the different parts of the system, the means for approving/disapproving modifications, and the level of permission each user has in terms of modifying others’ work. Essentially, this section defines the mechanisms and protocols set in place to control access to CIs in order to prevent unauthorized changes and inconsistencies between team members who are working on the system at the same time.Lastly, the software configuration auditing section describes the ways in which the degree of similarity between the current system and the proposed system will be measured. This section also contains the protocols set in place to ensure delivery of the correct product with all of the desired features.

The intended audience of this document are our clients, Dr. Oscar Perez, Vincent Fonseca, Herandy Denisse Vazquez, Baltazar Santaella, Florencia Larsen and Erick De Nava, as well as our guidance team composed of Dr. Gates, Dr. Salamah, Dr. Roach, Elsa Tai Ramirez and Peter Hanson.

## References

[1] Dr. Roach et al, Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK)

Software Requirements Specification, 2020.

# 2. Software Configuration Identification

This section defines conventions for versioning, databases used during development, document/code naming conventions as well as collaboration procedure, and back-up procedure.

## 2.1 Software Configuration Item Identification

<< Identify a complete list of elements that will make up a configuration. A “configuration” is the set of things you need to create and install a working version of your system. These items include source code, design documents, test suites, requirements documents, project plans, project standards, and other documentation such as user guides. It may also include specific versions of COTS products needed to build and run the system. Identify only the items that are to be controlled in a given configuration. >>

* Source Code
* Splunk
* Software Configuration Management Plan
* Testing files and data
* User Documentation

## 2.2 Software Configuration Item Organization

Each class documented in the SRS shall have its own folder for its source code and the program will be separated based on purpose such as UI ect. Class names shall use UpperCamelCase and capitalized with underscores for separators while lowercase with underscores will be used for everything else. Private will have underscores before the name.

Github will host the code and Eduardo Herrera will be in charge of backups and the times are still being decided depending if we can get the process automated. Backups shall be stored in a Google drive. Each teammate will work on their own machine and upload from it using git.

Each team member shall be responsible for pushing their code to git.

# 3. Software Configuration Control

In this section, we explain the process of approving or rejecting a change, how access to items will be determined, the steps that need to be followed, and how to evaluate the ramifications the change will cause

## 3.1 Documentation

<< Provide documentation for formally precipitating and defining a proposed change to a software system.

Explain how you will document changes to the configuration. What style will be used? NOTE: In a large software project, this section would include a series of forms or procedures for submitting a change request to a committee for review. A change request form usually contains information related to who is requesting the change, expected start and delivery dates, a description of the change, priority level, business justification for the change and a section to be completed by the development team where an initial assessment is provided on what the impact of the change will be in the system, level of effort needed to complete the change, approval signatures and actual start and delivery dates. This type of documentation assists project managers to maintain and evaluate metrics related to the progress of the project. >>

## 3.2 Configuration Control Board

For the PICK software system team MELJJ concluded that the Lead Programmer and the System Architect will be in charge of evaluating and approving any changes. They will be reviewed by them and confirm if the changes shall be made. The Lead Programmer and the System Architect will be responsible for changes and approval of different parts of the source code. However, remaining members will individually contribute to different parts of the code but will be working alongside the Lead Programmer and System Architect.

The V&V will report and record any errors in code by identifying the number of bugs created and the location where the malfunction occurred. All members of team MELJJ can make suggestions to any other members code however, all changes shall be pushed into individual git branches until either Lead Programmer or System Architect approves them and merge them into master branch.

## 3.3 Procedures

Jira would be used to define what are the tasks or features that each of the version is going to contain; this is going to be broken down into sprints. Furthermore, it would allow the team members to report any errors in any of the versions of the product. For creating new changes to configuration items we are going to use Git for version control. First step in the process is that each team member is going to be assigned with a Jira issue; the issue contains the version of the product and the feature that needs to be implemented. From there the team member is going to create a branch that pertains to that one issue, this is going to be the branch where the developer accomplishes the feature. Once the developer has successfully completed the feature test are going to be run in the feature, making sure that this new addition is going to break anything else in the system. Then the developer is ready for a pull request into the *development* *branch*; this branch captures all the issues for a specific version that is going to be released (the issues are defined in the Jira’s sprint). Another team member is going to review the code’s structure, potential vulnerabilities, and anti-patterns. Finally, if the this team member reviewing the code approves, the feature is merged into the *development branch*. Once all the features specified for a specific version of the system are merged in the *development branch*, the version is ready to go into production. The *development branch* is going to perform the final integration test and would be merged into master. The final step would be setting a tag in Github that specifies the version of the system that is in production.

# 4. Software Configuration Auditing

The mechanism for auditing items of our project will be using a checklist to make sure everything has been captured. This audit will ensure all procedures were followed and the item satisfies the specifications that it requires properly.

1. Are all the anomalies being reported during a verification and validation check?
2. Are anomalies being resolved?
3. Can each software requirement be traced forward?
4. Have all if any, changes been changed?
5. Have all software processes been followed and have all standards been applied?
6. Have all procedures been followed by the team?

$